ORCA: Operating system and Runtime system support for arChitectural diversity and Application dynamicity

Alok Choudhary (Northwestern), Mahmut Kandemir (Penn State), Wei-keng Liao (Northwestern)

(Contact Email: alokchoudhary01@gmail.com)

Emerging multicore systems are employing complex cache hierarchies, on-chip networks, multiple memory controllers, and on-chip heterogeneity (e.g., big core versus small core, and graphic processing support). This, coupled with emerging memory technologies such as PCRAM and STTRAM and storage technologies such as SSDs and hybrid drives, is changing the architectural spectrum and defining the computing platform of the near future. More importantly, two instantiations of this computing platform can be very different from each other. For example, one of them can include new nonvolatile memory technologies such as PCRAM and use 2D mesh as the on-chip interconnect, whereas another one can include DRAM and STTRAM and use hierarchical rings for inter-core communication. In parallel, high performance applications are becoming increasingly *data driven* where the main goal is optimizing data-to-discovery through summary statistics, query, reporting, analysis, as well as bottom-up hypothesis-driven and predictive discoveries. Clearly, existing operating systems (OS) and runtime systems are not designed to cope with this level of architectural diversity and application dynamicity, and therefore newer and innovative approaches are needed for the exascale regime.

We propose a novel OS/runtime support, *ORCA*, targeting emerging multicore-based systems and data driven and analytics applications. The distinguishing characteristics of our proposed approach include:

- In ORCA, *important architectural features will be "exposed" to the OS/runtime system.* As a result, the OS will have information about hardware features that are critical for system performance and power. These features will include the number and types of cores, on-chip network parameters (topology, bandwidth, etc), cache topology and parameters, the number of memory controllers and their positions on the chip, available GPU support, and memory/storage hierarchy details (including types of NVRAMs, capacities, management policies, etc). Exposing these parameters to the OS will enable it to carry out performance and pover optimizations that are not possible in the state-of-the-art.
- *Minimizing both on-chip and off-chip "data movement" is the" first-class optimization parameter" in ORCA.* To do this, the entire ORCA infrastructure will be designed with scalability in mind and with the principle that data and computation that operate on that data should be collocated. For example, when a certain thread accesses specific data pages in the off-chip memory, ORCA will bring that thread close to a memory controller (in the chip) and allocate the pages from the memory banks that are connected to that controller. This will reduce the distance that need to be travelled over the on-chip network to access off-chip data.
- *ORCA will target "end-to-end" performance-power optimization.* Consequently, in, say, partitioning resources (such as on-chip caches) among competing threads/applications, the goal will be to minimize end-to-end latency, instead of just minimizing cache misses (in fact, in a network based multicore with multiple memory controllers, different cache misses can experience very different latencies, depending on the contention they experience on the network as well as the queuing time they have in target memory banks). Note that exposing critical architectural characteristics to the OS is the key enabler for end-to-end optimization.
- *ORCA will exploit unique characteristics of different on-chip and off-chip hardware components.* For example, in a hybrid memory hierarchy with DRAMs, PCRAMs and STTRAMs, data pages will be dynamically migrated, depending on current data access patterns. For example, write-intensive pages (which will be identified at runtime) are not good candidates for PCRAM/STTRAM, as they incur higher latency and power (and can cause wear-out in PCRAM). Therefore, such PCRAM/STTRAM pages will be identified and migrated to DRAM. On the opposite side, pages that are not accessed for a long time (i.e., idle pages) can be migrated from DRAM to STTRAM as the former spends unnecessary refresh power for such pages.
- *IN ORCA, application threads and OS threads will be carefully co-scheduled considering interactions between them.* For example, if a group of threads use a certain OS function/service, an instantiation of that OS function will be brought close (in the chip) to those threads. This "colocation of application threads and OS threads" will not only minimize "distance-to-service", but it will also minimize "distance-to-data", as typically application

threads that exercise an OS service also share data with it. In addition, ORCA will employ customized scheduling support for components other than cores (e.g., memories).

Related Work

Two most related efforts to our proposal are NIXOS (http://nixos.org/nixos/) and Factored OS (http://groups.csail.mit.edu/carbon/). While Factored OS considers the issue of co-scheduling of application threads and OS threads, generally speaking, our proposed work is radically different from these efforts, as it exposes architectural details to the OS, treats "data movement" as the first-class objective to minimize, targets end-to-end optimization, and gives support for NVRAMs. In addition to these, there are some efforts that attempt to make Linux more resource aware, but these efforts are mainly targeting cache behavior, which we believe is only one part of the big picture in future exascale systems.

Assessment

Challenges addressed: The proposed approach is expected to address OS and application scalability, hardware and application level heterogeneity, power/performance cost of data movement, handling very large data sets, effective exploitation of emerging storage-class memories, and end-to-end performance/power optimization.

Maturity: First, our preliminary studies show that exposing architectural details to system software can bring significantly more power and performance improvements over what could be achieved using existing techniques alone. Second, we observed that, emerging nonvolatile memory technologies would be a viable option *only if* we have accompanying system software support customized for them (e.g., smart data-to-memory mapping and runtime data migration). Third, our experiments indicate that focusing on end-to-end performance optimization, instead of just considering cache hits/misses for example, has the potential to generate much higher improvements, primarily because, in emerging multicore based systems, neither hit latencies nor miss latencies are constant.

Uniqueness: ORCA will be implemented with two "scalability" concerns in mind: "OS /runtime system scalability" and "application scalability". In fact, *we believe that achieving OS/runtime scalability is the key to achieving application scalability in exascale systems.* ORCA has several components to address the scalability problem such as collocating data and computations that have affinity, minimizing both on-chip and off-chip data movements, and exploiting unique features of emerging memory and storage technologies.

Novelty: We are not aware of any existing OS/runtime system that employs optimizations to minimize data movement, exploit critical features of emerging storage-class memory as well as important architectural characteristics (such as memory queues, which is critical for data intensive applications), and target end-to-end power/performance optimization. Based on our initial assessment, we expect that ORCA will advance the state-of-the-art, leading to at least *4X performance improvement* and *6X energy improvement* in data-driven scientific applications and big data analytic kernels.

Applicability: If successful, ORCA will revolutionize the way we think about the OS and runtime system. In particular, the boundary between the underlying parallel computation/communication/storage fabric and the OS/runtime system will be *blurred*, allowing the latter to get *customized* at runtime based on the specific features of the former. Therefore, we expect our approach to be used in other environments where a more-coupled hardware-software co-optimization/interaction is strongly desired.

Effort: We already assessed the potential of exposing important hardware features to the runtime system, and quantified the benefits of targeting end-to-end performance optimization and dynamic data migration in hybrid memory systems. We expect one and half year to have a working prototype of ORCA, and another one and half year to have a robust implementation with all planned features.